

CLAIMS

What is claimed is:

- 1 1. A method of dynamic installation and activation of software packages in a node in a
2 distributed network of nodes, the method comprising the computer-implemented steps of:
3 providing a master node;
4 providing software package storage means on said master node for storing software
5 packages and software modules that the nodes in the network will be using as
6 well as older versions that are kept for regressing a node back to a previous
7 module or software package version;
8 wherein a software package contains at least one module and its associated
9 dependency information;
10 receiving a software update for a node on said master node;
11 wherein the software update contains a software package, or a set of software
12 packages;
13 storing the software update on said software package storage means;
14 wherein said master node notifies said node that a software update is being requested;
15 and
16 wherein said master node passes said node identities of software package(s) to be
17 updated and module dependencies.
- 1 2. A method as recited in Claim 1, wherein each module has a binary signature;.
- 1 3. A method as recited in Claim 1, wherein each node has a list of desired characteristics
2 stored on said master node which is compared by said master node to each module in the
3 software update to determine which subset of modules should be sent to a node.
- 1 4. A method as recited in Claim 1, wherein said node determines running processes on
2 said node that will be affected by the software update using the module dependencies.

1 5. A method as recited in Claim 4, wherein said node notifies affected processes that the
2 software update is being requested; wherein each notified process evaluates the effect that the
3 software update will have on its operation; wherein if any of the notified processes determine
4 that the software update will degrade or have a negative impact on said node's normal
5 operation, the process returns a veto to said node; and wherein if a process finds that the
6 software update will have no negative effects, the process returns an acceptance of the
7 software update to said node.

1 6. A method as recited in Claim 5, wherein said node waits for all of the notified
2 processes to return the results of their evaluations and once all of the processes have reported
3 to said node, said node notifies said master node if any of the processes have vetoed the
4 software update.

1 7. A method as recited in Claim 6, wherein if said master node receives an acceptance
2 from said node then said master node sends appropriate software package(s) for the software
3 update from said software package storage means to said node.

1 8. A method as recited in Claim 7, wherein said node immediately runs software
2 package modules, by loading the modules from the software package(s) and signals
3 processes that are being replaced by the modules and the affected processes that the
4 changeover is going to occur; wherein when all of the signaled processes indicate that they
5 are ready and waiting for the changeover, said node starts new modules and signals the
6 affected processes that the changeover has occurred; wherein each module starts without
7 affecting normal operation of said node; and wherein each affected process restarts, if
8 required, without affecting normal operation of said node.

1 9. A method as recited in Claim 8, wherein said node continues with normal operations
2 and notifies said master node that it has completed the software update; and wherein said
3 master node checks the dependencies of the software package(s) for the software update to
4 ensure that any inter-nodal and intra-node dependencies are complete.

1 10. A method as recited in Claim 9, wherein if there are any discrepancies in the inter-
2 nodal and intra-node dependencies, then said master node notifies a user.

1 11. A method as recited in Claim 8, wherein when said node does not store the software
2 package(s) in its local persistent storage, then said node can later regress back to previous
3 modules stored in the local persistent storage if it restarts or said master node tells it to
4 regress.

1 12. A method as recited in Claim 7, wherein said node extracts version information and
2 dependency information from the software package(s) and stores the information in its local
3 persistent storage.

1 13. A method as recited in Claim 12, wherein said node compares binary signatures of
2 the modules in the software package(s) with corresponding modules stored in the local
3 persistent storage to discover which modules have been updated; wherein any binary
4 signatures that match indicate that the module has not changed; and wherein any modules
5 that have different binary signatures replace the corresponding modules stored in the local
6 persistent storage.

1 14. A method as recited in Claim 13, wherein said node runs software package modules
2 by loading the modules from the software package(s) and signals processes that are being
3 replaced by the modules and the affected processes that the changeover is going to occur;
4 wherein when all of the signaled processes indicate that they are ready and waiting for the
5 changeover, the node starts new modules and signals the affected processes that the
6 changeover has occurred; wherein each module starts without affecting normal operation of
7 said node; and wherein each affected process restarts, if required, without affecting normal
8 operation of said node.

1 15. A method as recited in Claim 14, wherein said node continues with normal operations
2 and notifies said master node that it has completed the software update; and wherein said
3 master node checks the dependencies of the software package(s) for the software update to
4 ensure that any inter-nodal and intra-node dependencies are complete.

1 16. A method as recited in Claim 15, wherein if there are any discrepancies in the inter-
2 nodal and intra-node dependencies, then said master node notifies a user.

1 17. A method as recited in Claim 6, wherein if more than one node was being updated,
2 the software update will not occur if any node vetoes the software update.

1 18. A method as recited in Claim 6, wherein if said master node receives a veto from said
2 node, then said master node does not update said node and notifies a user that the software
3 update will adversely affect said node; and wherein the user must then make a decision
4 whether to update some or all of the nodes, or to abort the update.

1 19. A method as recited in Claim 18, wherein if the user decides to continue updating
2 said node, then said master node forces said node to accept the software update.

1 20. A method as recited in Claim 1, wherein a user initiates a software update by
2 installing an image containing the software update onto said master node.

1 21. A method as recited in Claim 20, wherein the user indicates what nodes and which
2 software package(s) are to be updated.

1 22. A method as recited in Claim 1, wherein the software update contains a list of nodes
2 to be updated.

1 23. A method as recited in Claim 1, wherein the software update contains a list of
2 software packages destined for each node.

1 24. A method as recited in Claim 1, wherein the master node has the ability to categorize
2 nodes into classes where all of the nodes in a particular class of nodes have the same
3 software configuration and may have differing processor types.

1 25. A method as recited in Claim 1, wherein a software package contains version
2 information, dependency information, and other metadata information pertaining to software
3 in the package.

1 26. A method as recited in Claim 25, wherein the metadata includes a list of application
2 program interface (API) providers and consumers.

1 27. A computer-readable medium carrying one or more sequences of instructions for
2 dynamic installation and activation of software packages in a node in a distributed network of
3 nodes, which instructions, when executed by one or more processors, cause the one or more
4 processors to carry out the steps of:
5 providing a master node;

6 providing software package storage means on said master node for storing software
7 packages and software modules that the nodes in the network will be using as
8 well as older versions that are kept for regressing a node back to a previous
9 module or software package version;
10 wherein a software package contains at least one module;
11 receiving a software update for a node on said master node;
12 wherein the software update contains a software package, or a set of software
13 packages;
14 storing the software update on said software package storage means;
15 wherein said master node notifies said node that a software update is being requested;
16 and
17 wherein said master node passes said node identities of software package(s) to be
18 updated and module dependencies.

1 28. A computer-readable medium as recited in Claim 27, wherein each module has a
2 binary signature;.

1 29. A computer-readable medium as recited in Claim 27, wherein each node has a list of
2 desired characteristics stored on said master node which is compared by said master node to
3 each module in the software update to determine which subset of modules should be sent to a
4 node.

1 30. A computer-readable medium as recited in Claim 27, wherein said node determines
2 running processes on said node that will be affected by the software update using the module
3 dependencies.

1 31. A computer-readable medium as recited in Claim 30, wherein said node notifies
2 affected processes that the software update is being requested; wherein each notified process
3 evaluates the effect that the software update will have on its operation; wherein if any of the
4 notified processes determine that the software update will degrade or have a negative impact
5 on said node's normal operation, the process returns a veto to said node; and wherein if a
6 process finds that the software update will have no negative effects, the process returns an
7 acceptance of the software update to said node.

1 32. A computer-readable medium as recited in Claim 31, wherein said node waits for all
2 of the notified processes to return the results of their evaluations and once all of the processes
3 have reported to said node, said node notifies said master node if any of the processes have
4 vetoed the software update.

1 33. A computer-readable medium as recited in Claim 32, wherein if said master node
2 receives an acceptance from said node then said master node sends appropriate software
3 package(s) for the software update from said software package storage means to said node.

1 34. A computer-readable medium as recited in Claim 33, wherein said node immediately
2 runs software package modules, by loading the modules from the software package(s) and
3 signals processes that are being replaced by the modules and the affected processes that the
4 changeover is going to occur; wherein when all of the signaled processes indicate that they
5 are ready and waiting for the changeover, the node starts new modules and signals the
6 affected processes that the changeover has occurred; wherein each module starts without
7 affecting normal operation of said node; and wherein each affected process restarts, if
8 required, without affecting normal operation of said node.

1 35. A computer-readable medium as recited in Claim 34, wherein said node continues
2 with normal operations and notifies said master node that it has completed the software
3 update; and wherein said master node checks the dependencies of the software package(s) for
4 the software update to ensure that any inter-nodal and intra-node dependencies are complete.

1 36. A computer-readable medium as recited in Claim 35, wherein if there are any
2 discrepancies in the inter-nodal and intra-node dependencies, then said master node notifies a
3 user.

1 37. A computer-readable medium as recited in Claim 34, wherein when said node does
2 not store the software package(s) in its local persistent storage, then said node can later
3 regress back to previous modules stored in the local persistent storage if it restarts or said
4 master node tells it to regress.

1 38. A computer-readable medium as recited in Claim 33, wherein said node extracts
2 version information and dependency information from the software package(s) and stores the
3 information in its local persistent storage.

1 39. A computer-readable medium as recited in Claim 38, wherein said node compares
2 binary signatures of the modules in the software package(s) with corresponding modules
3 stored in the local persistent storage to discover which modules have been updated; wherein
4 any binary signatures that match indicate that the module has not changed; and wherein any
5 modules that have different binary signatures replace the corresponding modules stored in the
6 local persistent storage.

1 40. A computer-readable medium as recited in Claim 39, wherein said node runs software
2 package modules by loading the modules from the software package(s) and signals processes
3 that are being replaced by the modules and the affected processes that the changeover is
4 going to occur; wherein when all of the signaled processes indicate that they are ready and
5 waiting for the changeover, the node starts new modules and signals the affected processes
6 that the changeover has occurred; wherein each module starts without affecting normal
7 operation of said node; and wherein each affected process restarts, if required, without
8 affecting normal operation of said node.

1 41. A computer-readable medium as recited in Claim 40, wherein said node continues
2 with normal operations and notifies said master node that it has completed the software
3 update; and wherein said master node checks the dependencies of the software package(s) for
4 the software update to ensure that any inter-nodal and intra-node dependencies are complete.

1 42. A computer-readable medium as recited in Claim 41, wherein if there are any
2 discrepancies in the inter-nodal and intra-node dependencies, then said master node notifies a
3 user.

1 43. A computer-readable medium as recited in Claim 32, wherein if more than one node
2 was being updated, the software update will not occur if any node vetoes the software update.

1 44. A computer-readable medium as recited in Claim 32, wherein if said master node
2 receives a veto from said node, then said master node does not update said node and notifies
3 a user that the software update will adversely affect said node; and wherein the user must
4 then make a decision whether to update some or all of the nodes, or to abort the update.

1 45. A computer-readable medium as recited in Claim 44, wherein if the user decides to
2 continue updating said node, then said master node forces said node to accept the software
3 update.

1 46. A computer-readable medium as recited in Claim 27, wherein a user initiates a
2 software update by installing an image containing the software update onto said master node.

1 47. A computer-readable medium as recited in Claim 46, wherein the user indicates what
2 nodes and which software package(s) are to be updated.

1 48. A computer-readable medium as recited in Claim 27, wherein the software update
2 contains a list of nodes to be updated.

1 49. A computer-readable medium as recited in Claim 27, wherein the software update
2 contains a list of software packages destined for each node.

1 50. A computer-readable medium as recited in Claim 27, wherein the master node has the
2 ability to categorize nodes into classes where all of the nodes in a particular class of nodes
3 have the same software configuration and may have differing processor types.

1 51. A computer-readable medium as recited in Claim 27, wherein a software package
2 contains version information, dependency information, and other metadata information
3 pertaining to software in the package.

1 52. A computer-readable medium as recited in Claim 51, wherein the metadata includes a
2 list of application program interface (API) providers and consumers

1 53. An apparatus of dynamic installation and activation of software packages in a node in
2 a distributed network of nodes, comprising:
3 a master node;

4 software package storage means on said master node for storing software packages
5 and software modules that the nodes in the network will be using as well as
6 older versions that are kept for regressing a node back to a previous module or
7 software package version;
8 wherein a software package contains at least one module;
9 means for receiving a software update for a node on said master node;
10 wherein the software update contains a software package, or a set of software
11 packages;
12 means for storing the software update on said software package storage means;
13 wherein said master node notifies said node that a software update is being requested;
14 and
15 wherein said master node passes said node identities of software package(s) to be
16 updated and module dependencies.

1 54. An apparatus as recited in Claim 53, wherein each module has a binary signature;.

1 55. An apparatus as recited in Claim 53, wherein each node has a list of desired
2 characteristics stored on said master node which is compared by said master node to each
3 module in the software update to determine which subset of modules should be sent to a
4 node.

1 56. An apparatus as recited in Claim 53, wherein said node determines running processes
2 on said node that will be affected by the software update using the module dependencies.

1 57. An apparatus as recited in Claim 56, wherein said node notifies affected processes
2 that the software update is being requested; wherein each notified process evaluates the effect
3 that the software update will have on its operation; wherein if any of the processes determine
4 that the software update will degrade or have a negative impact on said node's normal
5 operation, the process returns a veto to said node; and wherein if a process finds that the
6 software update will have no negative effects, the process returns an acceptance of the
7 software update to said node.

1 58. An apparatus as recited in Claim 57, wherein said node waits for all of the notified
2 processes to return the results of their evaluations and once all of the processes have reported
3 to said node, said node notifies said master node if any of the processes have vetoed the
4 software update.

1 59. An apparatus as recited in Claim 58, wherein if said master node receives an
2 acceptance from said node then said master node sends appropriate software package(s) for
3 the software update from said software package storage means to said node.

1 60. An apparatus as recited in Claim 59, wherein said node immediately runs software
2 package modules, by loading the modules from the software package(s) and signals
3 processes that are being replaced by the modules and the affected processes that the
4 changeover is going to occur; wherein when all of the signaled processes indicate that they
5 are ready and waiting for the changeover, the node starts new modules and signals the
6 affected processes that the changeover has occurred; wherein each module starts without
7 affecting normal operation of said node; and wherein each affected process restarts, if
8 required, without affecting normal operation of said node.

1 61. An apparatus as recited in Claim 60, wherein said node continues with normal
2 operations and notifies said master node that it has completed the software update; and
3 wherein said master node checks the dependencies of the software package(s) for the
4 software update to ensure that any inter-nodal and intra-node dependencies are complete.

1 62. An apparatus as recited in Claim 61, wherein if there are any discrepancies in the
2 inter-nodal and intra-node dependencies, then said master node notifies a user.

1 63. An apparatus as recited in Claim 60, wherein when said node does not store the
2 software package(s) in its local persistent storage, then said node can later regress back to
3 previous modules stored in the local persistent storage if it restarts or said master node tells it
4 to regress.

1 64. An apparatus as recited in Claim 59, wherein said node extracts version information
2 and dependency information from the software package(s) and stores the information in its
3 local persistent storage.

1 65. An apparatus as recited in Claim 64, wherein said node compares binary signatures of
2 the modules in the software package(s) with corresponding modules stored in the local
3 persistent storage to discover which modules have been updated; wherein any binary
4 signatures that match indicate that the module has not changed; and wherein any modules
5 that have different binary signatures replace the corresponding modules stored in the local
6 persistent storage.

1 66. An apparatus as recited in Claim 65, wherein said node runs software package
2 modules by loading the modules from the software package(s) and signals processes that are
3 being replaced by the modules and the affected processes that the changeover is going to
4 occur; wherein when all of the signaled processes indicate that they are ready and waiting for
5 the changeover, the node starts new modules and signals the affected processes that the
6 changeover has occurred; wherein each module starts without affecting normal operation of
7 said node; and wherein each affected process restarts, if required, without affecting normal
8 operation of said node.

1 67. An apparatus as recited in Claim 66, wherein said node continues with normal
2 operations and notifies said master node that it has completed the software update; and
3 wherein said master node checks the dependencies of the software package(s) for the
4 software update to ensure that any inter-nodal and intra-node dependencies are complete.

1 68. An apparatus as recited in Claim 67, wherein if there are any discrepancies in the
2 inter-nodal and intra-node dependencies, then said master node notifies a user.

1 69. An apparatus as recited in Claim 58, wherein if more than one node was being
2 updated, the software update will not occur if any node vetoes the software update.

1 70. An apparatus as recited in Claim 58, wherein if said master node receives a veto from
2 said node, then said master node does not update said node and notifies a user that the
3 software update will adversely affect said node; and wherein the user must then make a
4 decision whether to update some or all of the nodes, or to abort the update.

1 71. An apparatus as recited in Claim 70, wherein if the user decides to continue updating
2 said node, then said master node forces said node to accept the software update.

1 72. An apparatus as recited in Claim 53, wherein a user initiates a software update by
2 installing an image containing the software update onto said master node.

1 73. An apparatus as recited in Claim 72, wherein the user indicates what nodes and which
2 software package(s) are to be updated.

1 74. An apparatus as recited in Claim 53, wherein the software update contains a list of
2 nodes to be updated.

1 75. An apparatus as recited in Claim 53, wherein the software update contains a list of
2 software packages destined for each node.

1 76. An apparatus as recited in Claim 53, wherein the master node has the ability to
2 categorize nodes into classes where all of the nodes in a particular class of nodes have the
3 same software configuration and may have differing processor types.

1 77. An apparatus as recited in Claim 53, wherein a software package contains version
2 information, dependency information, and other metadata information pertaining to software
3 in the package.

1 78. An apparatus as recited in Claim 77, wherein the metadata includes a list of
2 application program interface (API) providers and consumers.